Kritik Assignment #4

1a.) Suppose $f:[a,b] \to \mathbb{R}$ is continuous and differentiable on (a,b) then there exists $c \in (a,b)$ such that

$$f'(c) = \frac{f(b) - f(a)}{b-a}$$

Explanation: As $f(x) = x^2 - 4x + 4$ is a polynomial and is not a rational function it can be determined that is continuous $\forall x \in \mathbb{R}$.

---

b) (1) Find the average rate of change btwn $[1,3]$

$$\text{avg} = \frac{f(3) - f(1)}{3-1}$$

$$= \frac{\left[(3)^2 - 4(3) + 4\right] - \left[(1)^2 - 4(1) + 4\right]}{2}$$

$$= 0/2$$

$$= 0$$

(2) Find points where instantaneous rate of change equals 0

$$f'(x) = 2x - 4$$

$$0 = 2x - 4$$

$$4 = 2x$$

$$\boxed{2 = x}$$

(3) $\therefore$ The point is $(2,0)$

$$y = x^2 - 4x + 4$$

$$y = 2^2 - 4(2) + 4$$

$$\boxed{y = 0}$$

---

c) As we have a polynomial that creates a parabola $x = 1$ and $x = 3$ correspond to the same y-value causing the average rate of change to be 0. This means the point $(2,0)$ is likely the absolute min of the parabola (a turning point) that is equidistant from $x = 1$ & $x = 3$ and has a derivative of 0.

2. $f(x) = \begin{cases} 3-\sqrt{x}, & x>0 \\ 3-\sqrt{-x}, & x<0 \end{cases}$

① Find Critical points

- $f'(c) = 0$
- $f'(c)$ d.n.e (doesn't exist)
- Boundaries of domain

Left Side

If $x<0$...

$f(x) = 3-\sqrt{-x}$

$f'(x) = \frac{1}{2\sqrt{-x}}$

As x will always be positive and greater than 0 there are no critical points from $(-4,0)$

Right Side

If $x \geq 0$

$f(x) = 3-\sqrt{x}$

$f'(x) = \frac{1}{2\sqrt{x}} \rightarrow x \neq 0$

The derivative D.N.E a zero making it a critical point

② Find absolute extrema

- Check at critical points
- Check at ends of domain

$f(0) = 3-\sqrt{0}$

$y = 3$

$f(-4) = 3-\sqrt{-(-4)}$

$= 3-2$

$= 1$

$f(4) = 3-\sqrt{4}$

$= 1$

∴ A absolute extrema occurs at $(0,3)$ & the abs minima occur at $(-4,1)$ & $(4,1)$

③ Find concavity (inflection)

From $(-4,0)$....

$f'(x) = \frac{1}{2\sqrt{-x}}$

$f''(x) =$ positive

∴ concave up

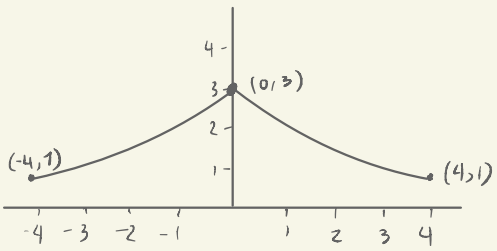
From $(0,4)$

$f'(x) = \frac{1}{2\sqrt{x}}$

$f''(x) =$ positive

∴ concave up

④ GRAPH



$f(x) = \ln(x^2+1)$

① Find Critical Points

$f'(x) = \frac{1}{x^2+1} \cdot 2x$

$= \frac{2x}{x^2+1}$

$0 = \frac{2x}{x^2+1}$

$0 = x$

∴ $f'(x) = 0$ at $x = 0$

★ $(-\infty, 0)$ the derivative is negative meaning function is decreasing. From $(0,\infty)$ the derivative is positive meaning function is increasing

② Find abs extrema

$f(0) = \ln(0^2+1)$

$= \ln(1)$

$= 0$

goes from decreasing to increasing ∴ there is a absolute min at $(0,0)$

③ Find Concavity / Points of Inflection

$f'(x) = \frac{2x}{x^2+1}$

$f''(x) = \frac{2(x^2+1) - 2x(2x)}{(x^2+1)^2}$

$0 = \frac{2(x^2+1) - 4x^2}{(x^2+1)^2}$

$-2 = 2x^2 - 4x^2 + 2$

$-2 = -2x^2$

$2 = 2x^2$

$\pm 1 = x$

↓

∴ There are points of inflection at 1 & -1

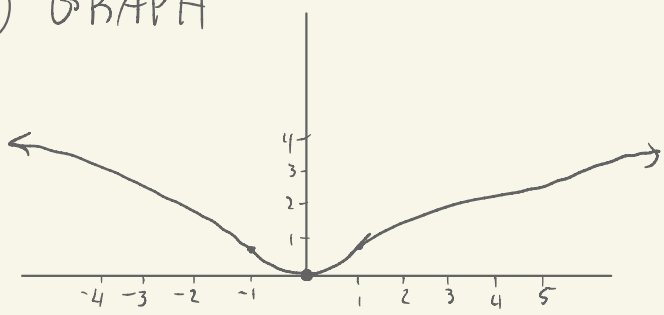
$f(1) = \ln(1^2+1)$

$= 0.69$

$f(-1) = 0.69$

	$(-\infty, -1)$	$(-1, 1)$	$(1, \infty)$
$f''(x)$	−	+	−
$f(x)$	∩	∪	∩

④ GRAPH



$f(x) = \frac{e^x + e^{-x}}{2}$

① Critical Points

$f'(x) = \frac{2e^x - 2e^{-x}}{4}$

$0 = e^x - e^{-x}/2$

$0 = e^x - e^{-x}$

$= x \ln e + x \ln e$

$= 2x$

∴ Critical point at $x = 0$ and $y = 1$ $(0, 1)$

↳ $f(0) = \frac{e^0 + e^{-0}}{2}$

$= \frac{1+1}{2}$

$= 1$

② Find abs extrema

• Critical point at $(0, 1)$

	$(-\infty, 0)$	$(0, \infty)$
$f'(x)$	−	+
$f(x)$	↘	↗

∴ Absolute minimum at $(0, 1)$

③ Find Concavity & Points of Inflection

$$f'(x) = \frac{e^x - e^{-x}}{2}$$

$$f''(x) = \frac{(e^x + e^{-x}) \cdot 2}{2^2}$$

$$0 = \frac{e^x + e^{-x}}{2}$$

→ $e^x + e^{-x}$ is always 1

$$0 = e^x + e^{-x}$$

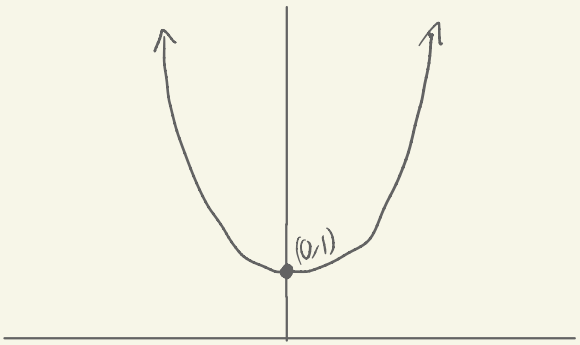
$$0 = x - x$$

$$= 0$$

★ NO points of inflection

(no x-value corresponds to a second derivative of zero)

④ GRAPH



3. $\lim_{x \to 0} \frac{1 - \cos(x)}{x^2}$ → when 0 is subed in we get $\frac{1 - \cos(0)}{0^2} = \frac{1-1}{0} = \frac{0}{0}$ Type 0

∴ As this is of type 0/0 we can use L'Hôpitals rule which states $\lim_{x \to a} \frac{f(x)}{g(x)} = \lim_{x \to a} \frac{f'(x)}{g'(x)}$

$= \lim_{x \to 0} \frac{\sin x}{2x}$ → type 0/0

$= \lim_{x \to 0} \frac{\cos x}{2}$

$= 1/2$

$\lim_{x \to \infty} \frac{e^{2x} + \frac{3}{x}}{e^{2x} + \frac{4}{x}}$ → type $\frac{\infty}{\infty}$

$= \lim_{x \to \infty} \frac{x\left(e^{2x} + \frac{3}{x}\right)}{x\left(e^{2x} + \frac{4}{x}\right)}$

$= \lim_{x \to \infty} \frac{xe^{2x} + 3}{xe^{2x} + 4}$ ← type $\frac{\infty}{\infty}$

$= \lim_{x \to \infty} \frac{e^{2x} + 2xe^{2x}}{e^{2x} + 2xe^{2x}}$

$= 1$

$$\lim_{x\to 0} \frac{4^x - 3^x}{x^3 - x^2} \rightarrow \frac{1-1}{0} = \frac{0}{0}$$ ← type 0/0 ∴ can use L'Hôpital's rule

$$= \lim_{x\to 0} \frac{4^x \ln 4 - 3^x \cdot \ln 3}{3x^2 - 2x}$$

↳ bottom is zero but top is not this means L'Hôpitals rule can no longer be applied as we have a $\frac{\#}{0}$

∴ The limit does not exist as $x \to 0$

$$\lim_{x\to\infty} \left(\sqrt{x^2 + x} - x\right)$$

$$= \lim_{x\to\infty} \frac{\sqrt{x^2+x} - x}{1} \cdot \left(\frac{\sqrt{x^2+x} + x}{\sqrt{x^2+x} + x}\right)$$

$$= \lim_{x\to\infty} \frac{x^2 + x - x^2}{\sqrt{x^2+x} + x}$$

$$= \lim_{x\to\infty} \frac{x}{\sqrt{x^2+x} + x}$$

$$= \lim_{x\to\infty} \frac{x/x}{\sqrt{x^2+x}/x + x/x}$$

$$= \lim_{x\to\infty} \frac{1}{\frac{x}{x} + \frac{x^{1/2}}{x} + \frac{x}{x}}$$

$$= \lim_{x\to\infty} \frac{1}{1 + \frac{1}{\sqrt{x}} + 1}$$

as $x \to \infty$ this # is getting infinitely close to 0

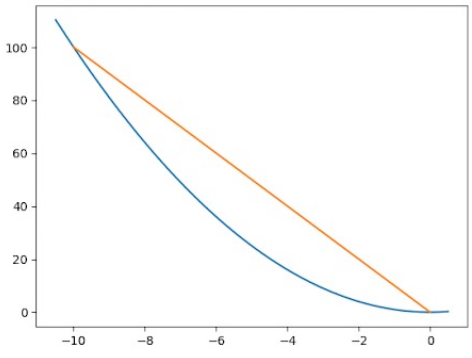
$$= \frac{1}{2}$$

4. See python below

$f(x) = x^2$

```python
import matplotlib.pyplot as plt
import numpy as np

def gradient_decent(f, learning_rate, base_point): #initial_point is the x_plot

    def deriv(f, base_point): #takes in function and base point (point u want to find derivative at)
        return (f(base_point+10**(-5))-f(base_point))/10**(-5) #definition of a dervivative in python (takes a rlly small # rather than a limit)
    x_coords=[base_point]
    y_coords=[f(base_point)] #first value we have in our list

    #we want to first calculate our first linear approximation and check to see when we get an error tolerance of rlly small #
    point = base_point - learning_rate * deriv(f,base_point) #initialize first linear approx with Xo
    #x1 = initial_point - learning_rate * deriv(f,initial_point)

    for i in range(100):

        x_coords.append(point)
        y_coords.append(f(point))

        point = point - learning_rate * deriv(f,point) #each time we run the loop the point will change and get closer to min

        if abs(deriv(f,point))<0.001:
            break

    if abs(deriv(f,point))>=0.001:
        print('there is no solution')

    plot_range=np.linspace(min(x_coords)-0.5, max(x_coords)+0.5,10000) #code given to us by Greg
    function_range=[f(i) for i in plot_range]
    plt.plot(plot_range, function_range)
    plt.plot(x_coords, y_coords)

    return round(x_coords[-1],3), round(y_coords[-1],3)

def f(x): #Input function that will run through the code
    return x**2

base_point = -10
learning_rate = 0.5
result = gradient_decent(f,learning_rate,base_point) #To print  the x and y point the minimum occurs at
print('the min occurs at',result)
```

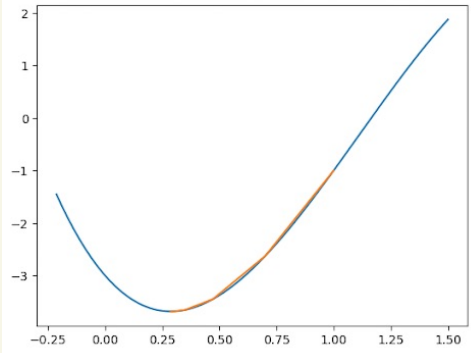
```
the min occurs at (-0.0, 0.0)
```



$f(x) = -3x^3 + 10x^2 - 5x - 3$

```python
import matplotlib.pyplot as plt
import numpy as np

def gradient_decent(f, learning_rate, base_point): #initial_point is the x_plot

    def deriv(f, base_point): #takes in function and base point (point u want to find derivative at)
        return (f(base_point+10**(-5))-f(base_point))/10**(-5) #definition of a dervivative in python (takes a rlly small # rather than a limit)
    x_coords=[base_point]
    y_coords=[f(base_point)] #first value we have in our list

    #we want to first calculate our first linear approximation and check to see when we get an error tolerance of rlly small #
    point = base_point - learning_rate * deriv(f,base_point) #initialize first linear approx with Xo
    #x1 = initial_point - learning_rate * deriv(f,initial_point)

    for i in range(1000):

        x_coords.append(point)
        y_coords.append(f(point))

        point = point - learning_rate * deriv(f,point) #each time we run the loop the point will change and get closer to min

        if abs(deriv(f,point))<0.001:
            break

    if abs(deriv(f,point))>=0.001:
        print('there is no solution')

    plot_range=np.linspace(min(x_coords)-0.5, max(x_coords)+0.5,10000) #code given to us by Greg
    function_range=[f(i) for i in plot_range]
    plt.plot(plot_range, function_range)
    plt.plot(x_coords, y_coords)

    return round(x_coords[-1],3), round(y_coords[-1],3)

def f(x): #Input function that will run through the code
    return -3*(x**3)+10*(x**2)-5*x-3

base_point = 1
learning_rate = 0.05
result = gradient_decent(f,learning_rate,base_point) #To print  the x and y point the minimum occurs at
print('the min occurs at',result)
```

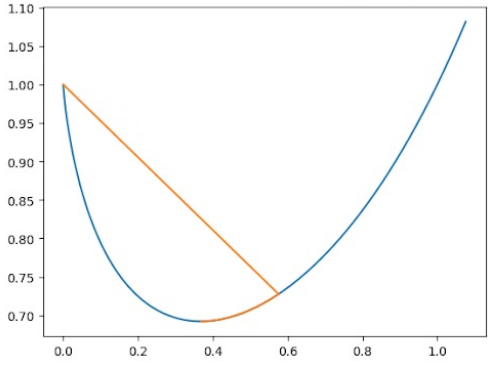
```
the min occurs at (0.287, -3.682)
```



$$f(x) = x^x$$

```python
import matplotlib.pyplot as plt
import numpy as np

def gradient_decent(f, learning_rate, base_point): #initial_point is the x_plot

    def deriv(f, base_point): #takes in function and base point (point u want to find derivative at)
        return (f(base_point+10**(-5))-f(base_point))/10**(-5) #definition of a dervivative in python (takes a rlly small # rather than a limit)
    x_coords=[base_point]
    y_coords=[f(base_point)] #first value we have in our list

    #we want to first calculate our first linear approximation and check to see when we get an error tolerance of rlly small #
    point = base_point - learning_rate * deriv(f,base_point) #initialize first linear approx with Xo
    #x1 = initial_point - learning_rate * deriv(f,initial_point)

    for i in range(1000):

        x_coords.append(point)
        y_coords.append(f(point))

        point = point - learning_rate * deriv(f,point) #each time we run the loop the point will change and get closer to min

        if abs(deriv(f,point))<0.001:
            break

    if abs(deriv(f,point))>=0.001:
        print('there is no solution')

    plot_range=np.linspace(min(x_coords)-0.5, max(x_coords)+0.5,10000) #code given to us by Greg
    function_range=[f(i) for i in plot_range]
    plt.plot(plot_range, function_range)
    plt.plot(x_coords, y_coords)

    return round(x_coords[-1],3), round(y_coords[-1],3)

def f(x): #Input function that will run through the code
    return x**x

base_point = 0
learning_rate = 0.05
result = gradient_decent(f,learning_rate,base_point) #To print  the x and y point the minimum occurs at
print('the min occurs at',result)
```

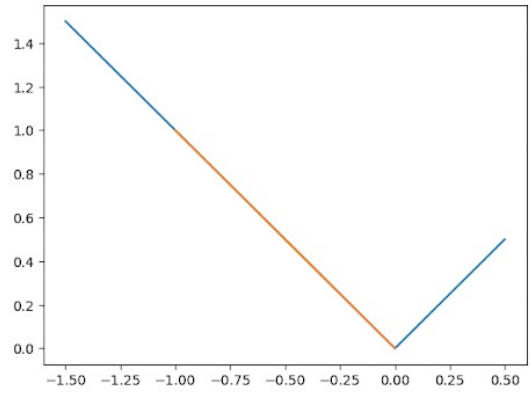
```
the min occurs at (0.368, 0.692)
```



$f(x) = |x|$

```python
import matplotlib.pyplot as plt
import numpy as np

def gradient_decent(f, learning_rate, base_point): #initial_point is the x_plot

    def deriv(f, base_point): #takes in function and base point (point u want to find derivative at)
        return (f(base_point+10**(-5))-f(base_point))/10**(-5) #definition of a dervivative in python (takes a rlly small # rather than a limit)
    x_coords=[base_point]
    y_coords=[f(base_point)] #first value we have in our list

    #we want to first calculate our first linear approximation and check to see when we get an error tolerance of rlly small #
    point = base_point - learning_rate * deriv(f,base_point) #initialize first linear approx with Xo
    #x1 = initial_point - learning_rate * deriv(f,initial_point)

    for i in range(1000):

        x_coords.append(point)
        y_coords.append(f(point))

        point = point - learning_rate * deriv(f,point) #each time we run the loop the point will change and get closer to min

        if abs(deriv(f,point))<0.001:
            break

    if abs(deriv(f,point))>=0.001:
        print('there is no solution')

    plot_range=np.linspace(min(x_coords)-0.5, max(x_coords)+0.5,10000) #code given to us by Greg
    function_range=[f(i) for i in plot_range]
    plt.plot(plot_range, function_range)
    plt.plot(x_coords, y_coords)

    return round(x_coords[-1],3), round(y_coords[-1],3)

def f(x): #Input function that will run through the code
    return abs(x)

base_point = -1
learning_rate = 0.05
result = gradient_decent(f,learning_rate,base_point) #To print  the x and y point the minimum occurs at
print('the min occurs at',result)
```

```
there is no solution
the min occurs at (0.0, 0.0)
```



Explanation: Gradient decent doesn't work with the function $f(x) = |x|$ as the derivative at (0,0) point does not exist and it is therefore unable to find a point where the derivative equals zero (min point)